DOI: 10.3745/KIPSTD.2009.16-D.4.549

애자일 기법을 이용한 소프트웨어 분산 개발 및 평가

이 세 영[†] · 용 환 승[††]

요 약

본 논문에서는 글로벌 IT 업계의 현실적인 대안으로 부상하고 있는 분산 스크럼 전략과 최적화된 애자일 기법을 사용하여 애자일 소프트웨어 분산 개발 프레임워크(AFDSD)를 제안하였다. AFDSD를 미국 야후사의 카멜레온 프로젝트에 실제 적용하고 종합적인 평가를 수행한 결과, 그 성능과 만족도가 30% 이상 향상되었다. 또한, 개발 조직의 애자일 도입 수준 평가, 프로세스와 프랙티스의 민첩도 산출, 애자일 프로젝트 성공도 산출 및 이전 버전과의 생산성과 품질 비교를 수행함으로써 애자일 프로젝트에 대한 새로운 평가 모델을 제시하였다. 본 연구의 목적은 실제 성공적인 적용사례를 통해 그 효율성이 검증된 분산 애자일 전략을 반영한 소프트웨어 개발 및 평가 모델을 제공함으로써 일반 업체들이 각자의 프로젝트 환경에 맞게 변형하여 손쉽게 적용 가능하도록 하는 것이다.

키워드 : 애자일 방법론, 스크럼, 소프트웨어 분산 개발, 애자일 프로젝트 평가

Distributed Development and Evaluation of Software using Agile Techniques

Lee, Seiyoung[†] · Yong, Hwan-Seung[††]

ABSTRACT

The Agile movement is a phenomenon that is part of the next phase of the software engineering evolution. At the same time, globally distributed software development is another trend delivering high-quality software to global users at lower costs. In this paper, Agile Framework for Distributed Software Development (AFDSD) has been suggested, and Chameleon project of Yahoo! Inc. has been implemented based on the framework. Also, the project has been evaluated by measuring Agile adoption and improvement levels, degrees of agility and agile project success, and comparing the performance and quality with the previous version. The overall performance and satisfaction with Chameleon increased by more than 30% since Agile techniques were adopted. Our objective is to highlight successful practices and suggest a framework to support adoption and evaluation of Agile techniques in a distributed environment.

Keywords : Agile Methods, Scrum, Distributed Software Development, Evaluation of Agile Projects

1. 서 론

소프트웨어 위기란 컴퓨터 하드웨어의 급속한 발전과 컴퓨터의 대중화로 인한 급격한 소프트웨어의 수요에도 불구하고 소프트웨어의 생산성과 생산 기술은 그에 미치지 못함으로써 나타난 현상으로 해석된다. 이에 이를 극복하기 위한 소프트웨어 공학 분야의 한현상으로 애자일(Agile) 소프트웨어 개발 방법론이 전 세계 IT 업계에 빠르게 확산[1, 2]되고 있다. 포레스터 리서치의 2008년 2월 보고서[3]에 따르면, 북미와 유럽의 약 1/4 가량의 기업이 애자일을 이미 도입했고, 절반 가량 기업의 대다수가 이미 도입 계획을 가지고 있거나 검토 중에 있다. 애자일 도입률은 2006년과 2007년 사이에 53% 증가하였으며, 기업의 규모가 클수록 애자일을 더욱 많이 사용하는 경향이 있는 것으로 나타났다. 외국의 경우 1990년대 북미와 유럽을 시작으로 애자일에 관한 방대한 양의 연구 및 실제 적용이 활발하게 진행되어 그 괄목할만한 효과가 입증[1-3]되고 있다. 반면, 국내에서는 2000년대 초반부터 주로 개발자 및 관련 커뮤니티에서 적용성이 검토되기 시작하여 아직까지 활발한 연구가 진행되고 있지 못하고 있는 실정이다[4, 5].

한편, 소프트웨어 분산 개발은 고품질의 소프트웨어를 최소한의 비용을 들여 생산하고자 하는 노력의 일환으로, 아웃 소싱을 넘어 글로벌 소싱 시대로 접어든 현재 필수적인 대안으로 받아들여지고 있다[6-8]. 이에 따라, 2008년 애자일 컨퍼런스에서는 분산 애자일(Distributed Agile)을 하나의 세션으로 오픈하여 전 세계 실무 개발자들의 현장 경험이 집중적으로 논의되었다[9]. 이처럼 애자일 소프트웨어 분산 개발은 글로벌 IT 업체로 하여금 저비용, 고효율이라는 두

† 정 회 원 : 이화여자대학교 컴퓨터공학과 박사과정
†† 종신회원 : 이화여자대학교 컴퓨터공학과 교수
논문접수 : 2009년 3월 24일
수 정 일 : 1차 2009년 5월 6일, 2차 2009년 5월 12일
심사완료 : 2009년 5월 15일

마리 토끼를 잡을 수 있는 기회를 제공하고 있다.

실제 많은 산업현장에서 애자일 방법론의 도입에 따른 효과는 막대하다[1-3, 9-13]. 본 연구에서는 전통적인 폭포수 방식 프로세스의 한계를 극복하기 위해 애자일이 도입된 기존사례를 종합적으로 분석, 검토하였다. 특히, 분산된 개발환경에서 애자일 프로젝트를 집중 연구하여 이를 바탕으로 애자일 분산 소프트웨어 개발 프레임워크(Agile Framework for Distributed Software Development: AFDSD)를 설계하였다. 이에 기초하여 미국 야후사의 카멜레온 프로젝트를 성공적으로 구현하였으며, 애자일 프로젝트에 대한 최신 평가 기술을 적용하여 이를 종합적으로 평가하였다. 그 결과, 개발 프로세스 및 생산성 전반에 걸쳐 약 30% 이상의 향상 효과를 볼 수 있었다. 본 논문의 구성은 2장에서 애자일 소프트웨어 분산 개발에 관한 기존 연구 결과를 분석하였고, 3장에서 이를 기반으로 AFDSD를 설계하였으며, 4장에서 AFDSD의 적용사례를 실제 구현하였다. 5장에서는 AFDSD의 적용사례를 종합적으로 평가 및 논의하였으며, 끝으로 6장에서 본 연구의 의의 및 요약을 기술하였다.

2. 관련 연구

프로젝트 실패의 근본적인 원인은 대부분 잘못된 커뮤니케이션에서 출발한다[6]. 이러한 전제 하에 대면 커뮤니케이션을 포함한 직접적인 상호작용을 중시하는 애자일 방법론과 이에 대한 구조적인 어려움이 따르는 소프트웨어 분산 개발은 태생적으로 대치된다. 애자일 방법론은 커뮤니케이션, 조정, 관리 등을 원활히 하기 위해 형식과 절차를 중시하지 않는다. 반면, 분산 소프트웨어 개발은 지리적, 문화적 장애를 극복하기 위해 전형적으로 형식적인 절차에 의존하는 경향이 있다[8].

2.1 애자일 방법론

대부분의 애자일 방법론은 산업 현장에서 실제 프로젝트를 진행하는 과정에서 필요에 의해 개발되었다. 예를 들어, 1990년대 미국을 거점으로 개발된 크리스탈(Crystal), 스크럼(Scrum), FDD(Feature-Driven Development), XP(Extreme Programming)는 각각 IBM, Easel, Nebulon, Chrysler사에서 시작되었고, DSDM(Dynamic Systems Development Method)만이 유일하게 영국에서 상업용 소프트웨어 개발 관리를 위한 산학 공동 연구의 결과로 탄생하였다[14]. 이들 방법론의 창시자들은 방대한 양의 문서와 산출물 기반의 전통적인 소프트웨어 개발 프로세스를 대신할 대안을 찾고 있다는 점에서 그 뜻을 같이 하였다. 이에 이러한 방법론들을 2001년 "애자일(Agile)"이라 통칭하고 다음과 같은 애자일 선언문(Agile Manifesto)을 공표하였다[15].

- 적극적인 고객과의 협력을 계약 협상보다 우선시 한다.
- 실제 동작하는 소프트웨어를 광범위한 문서보다 우선시 한다.
- 개개인 간의 소통을 프로세스와 도구보다 우선시 한다.
- 변화에 대응한 행동을 계획의 수행보다 우선시 한다.

즉, 프로젝트 초기부터 고객의 적극적인 참여를 유도하고, 단기간의 반복/점증적인 제품 출시를 통해 시장의 반응을 빠르게 적용하며, 전문성이 높은 소규모의 팀을 꾸려 효과적인 대화와 협력을 통해 생산성을 극대화한다. 이처럼 변화무쌍한 시장과 고객의 요구 변화에 유연하게 대응함으로써 고객의 만족과 비즈니스의 성공을 이끌어내는 데에 그 핵심이 있다. 애자일이라는 범주 아래 존재하는 여러 방법론은 용어 및 그 구체적인 실행 방법은 달리하나 비전과 목표의 공유, 자발적인 팀, 상식, 고품질, 지속적인 개선이라는 공통의 가치를 지향한다.

2.2 애자일 소프트웨어 분산 개발

분산 개발팀이 나날이 직면하고 있는 거리, 시간, 문화, 언어 장벽 및 조직적, 기능적 한계에도 불구하고[7], 글로벌 소싱과 소프트웨어 분산 개발은 글로벌 대기업에게 선택이 아닌 필수로 자리잡고 있다[16]. 개발 비용감소, 지역별 근무시간대 적용(follow-the-sun)을 통한 개발주기 감소, 개발작업의 지역별 모듈화, 대량의 고급 개발 인력, 기술 혁신, 최고의 프랙티스 공유, 지역 고객과의 근접 효과 등 글로벌 분산개발을 통해 기업은 많은 잠재적 이득을 취할수 있기 때문이다[6].

성공적인 글로벌 소프트웨어 분산 개발에 대한 관심이 고조될 무렵, 애자일 방법론은 예산 초과, 일정지연, 기대에 미치지 못하는 품질 등 소프트웨어 개발의 고질적인 문제를 다루기 위한 새로운 패러다임으로 제안되었다[6]. Ramesh 등[8]은 분산 개발에 애자일을 적용함에 있어 발생 가능한 5가지 도전으로써 1) 커뮤니케이션의 필요성 vs. 장벽, 2) 고정된 품질 요구사항 vs. 진화하는 요구사항, 3) 사람 vs. 프로세스 지향적인 관리, 4) 형식적 vs. 비형식적 절차, 5) 팀 응집력의 결여 등을 꼽았다. 더불어 애자일 분산 개발을 성공시키기 위해 다음과 같은 5가지 프랙티스를 제안하였다. - 1) 지속적으로 프로세스를 조정해나간다, 2) 지식의 공유를 원활히 한다, 3) 커뮤니케이션을 개선한다, 4) 신뢰를 구축한다, 5) 신뢰하되 검증한다. 한편, Simons[6]는 빈약한 문서화에 대한 부분 등을 효과적으로 보완하면 애자일 기법은 기존의 어떠한 방법론보다 분산 환경에서의 커뮤니케이션 문제를 효과적으로 잘 다루고 있다고 주장했다. 또한, Lee 등[7]은 글로벌 분산 환경에서 애자일 소프트웨어 개발을 성공적으로 수행하기 위해 유연성과 엄격한 규율을 동시에 수용하도록 순수 애자일 기법을 확장해야 한다고 지적하고 이를 위한 구체적인 전략을 제안하였다.

2.3 분산 스크럼의 실현 가능성

스크럼의 창시자인 Sutherland 등[13]은 스크럼과 XP의 효과적인 결합[11, 17]을 통해 생산성을 5~10배 향상시킬 수 있는 강력한 개발팀을 구성하였다. 또한, 1993년 스크럼

이 탄생한 이래 많은 일반 스크럼(동일 장소에서 협업하는 스크럼)팀이 이러한 주장을 뒷받침해오고 있다고 주장했다. 또한, Sutherland 등[12]은 글로벌 분산 환경에서 스크럼이 일반 스크럼과 같은 수준의 개발 속도를 가져올수 있는지를 증명하기 위해 2005년, 미국의 SirsiDynix사, 러시아의 Exigen Services사와 더불어 분산 스크럼 팀을 구성하여 1M 라인 이상의 코드를 개발하는 대규모 프로젝트를 수행하였다. 미국과 러시아로부터 50명 이상의 인원이 투입된 이 프로젝트는 분산 환경임에도 불구하고 일반 스크럼 팀과 같은 수준의 개발 속도를 내는데 성공하였다. Sutherland 등[13]은 다시 [12]에서 증명한 분산 스크럼의 성능이 지속적으로 다양한 유형의 프로젝트 환경에서 실현 가능한지를 입증하기 위해, 2006년부터 2008년에 걸쳐, Xebia사와 더불어 소프트웨어 분산 개발 팀 모델을 구성하여 다양한 유형의 프로젝트를 수행하였다. 이러한 분산 개발팀의 절반은 네덜란드, 나머지 절반은 인도를 거점으로 근무하였고, 스크럼 프로세스에 XP 엔지니어링 프렉티스를 결합하여 사용하였다. 결과적으로 여러 프로젝트를 수행하면서 측정된 개발 속도는 평균적으로 SirsiDynix사가 구현했던 분산 스크럼의 경우와 동일한 수준이었다. 즉, 약 4년에 걸친 이 연구는 분산 혹은 아웃 소싱 스크럼이 일반 스크럼과 같은 수준의 개발 속도를 낼 수 있으며, 다양한 유형의 프로젝트에 대해 반복적으로 실현 가능함을 보였다.

2.4 애자일 프로젝트 평가

Chow와 Cao[21]는 애자일 프로젝트의 성공 여부를 결정하는 6가지 주요 요소 및 그 속성을 제안하였다. 25개국으로부터 109개의 애자일 프로젝트로부터 얻은 데이터를 복합회귀 기법을 사용, 분석하여 프로젝트 성공을 결정하는 품질, 범위, 시간, 비용의 4가지 차원 각각에 대해 12가지의 성공요소에 대한 가설을 세우고 정량적인 분석을 수행하였다. 그 결과, 제품 출시 전략, 애자일 소프트웨어 엔지니어링 기술, 팀 역량 등이 애자일 프로젝트의 성공을 좌우하는 핵심 요소임이 발견되었다<표 3>. 한편, Qumer와 Henderson-Sellers[22]는 산업계의 다양한 적용사례와 이론을 바탕으로 애자일 도입 및 개선 모델을 제안하였다. 여기에 포함된 총 6개의 애자일 도입 및 개선 단계 (Agile Adoption and Improvement Levels: AAIML)는 새로이 애자일 방법론을 도입하는 업체의 경우 로드맵으로 이용할 수 있으며, 현재 애자일을 사용하고 있는 업체는 해당 조직의 애자일 수준을 평가 및 개선하는 도구로 활용할 수 있다(그림 7). 또한, Qumer와 Henderson-Sellers[23]는 소프트웨어 개발 방법론의 민첩도 평가를 위한 범용 프레임워크로써 4-DAT(4-Dimensional Analytical Tool)를 제안하고, 이를 적용하여 기존의 주요 애자일 방법론과 전통적인 방법론의 민첩도(Degree of Agility)를 비교 평가하였다. 이 도구는 방법론의 범위, 민첩도의 속성, 애자일 가치, 프로세스의 4가지 차원에 근거하여 민첩도를 분석한다.

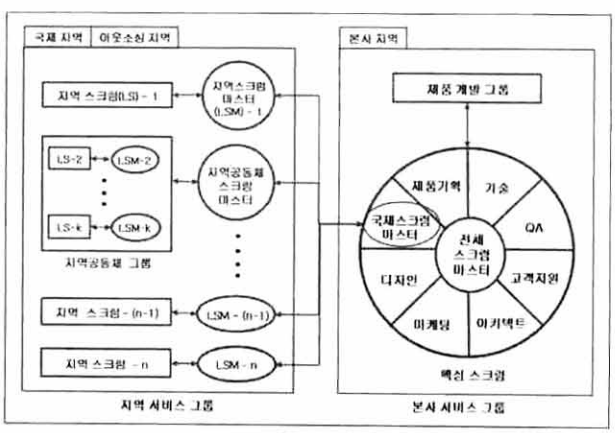
3. 애자일 소프트웨어 분산 개발 프레임워크 (AFDSD) 설계

본 논문에서 제안한 AFDSD는 소프트웨어 분산 개발 환경에 적합하도록 분산 스크럼 전략과 민첩도가 높은 애자일 프랙티스를 결합한 것이 그 핵심이다. AFDSD는 분산 스크럼 기반의 개발 조직 모델, 소프트웨어 개발 라이프 사이클을 반영한 스크럼 프로세스, 그리고 분산 개발환경에 최적화된 프랙티스로 구성된다.

3.1 AFDSD 개발 조직 모델

본 절에서는 소프트웨어 분산 개발을 효과적으로 수행하기 위한 개발 조직 모델 및 스크럼 전략을 기술한다. AFDSD는 본사의 서비스 그룹을 크게 핵심팀과 제품 개발 그룹으로 구분한다(그림 1). 핵심팀은 제품 기획, 기술, QA, 마케팅, 디자인, 고객 지원팀의 대표와 담당 아키텍트, 프로그램 매니저 및 국제 버전 담당자 등으로 구성되어 각 주관 업무에 관한 소속팀의 의사 결정을 주도한다. 제품 개발 그룹에는 제품의 기획부터 디자인, 설계, 개발, QA, 서비스 운영에 이르기까지 다양한 업무를 수행하는 팀이 공존하고 있으며 필요에 따라 아웃 소싱을 활용할 수 있다. 이처럼 본사 주도하에 여러 지역팀이 협업하는 경우, 본사에 국제 버전 담당자를 두어 각 지역팀과의 협업을 통해 지역 서비스를 중앙에서 관리하거나, 필요에 따라 중간에 지역 공동체 관리자를 두어 보다 계층적인 관리를 수행할 수 있다. 이러한 조직 모델을 스크럼에 적용하면 (그림 1)과 같이 각 관할 조직의 프로덕트 혹은 프로젝트 매니저가 스크럼 마스터 역할을 맡아 분산 스크럼을 구성할 수 있다.

AFDSD는 글로벌 환경에 유연하게 대처하기 위해 Sutherland[13]가 정의한 아래 세가지 유형의 분산 스크럼을 선택적으로 수용한다. 적정 수준의 독립성이 보장되면서도 스크럼으로 연결되어 그 이점을 취할 수 있는 분산 스크럼의 스크럼은 스크럼 연합이 추천하는 분산 스크럼 모델이



(그림 1) 분산 스크럼 기반의 AFDSD 개발 조직 모델

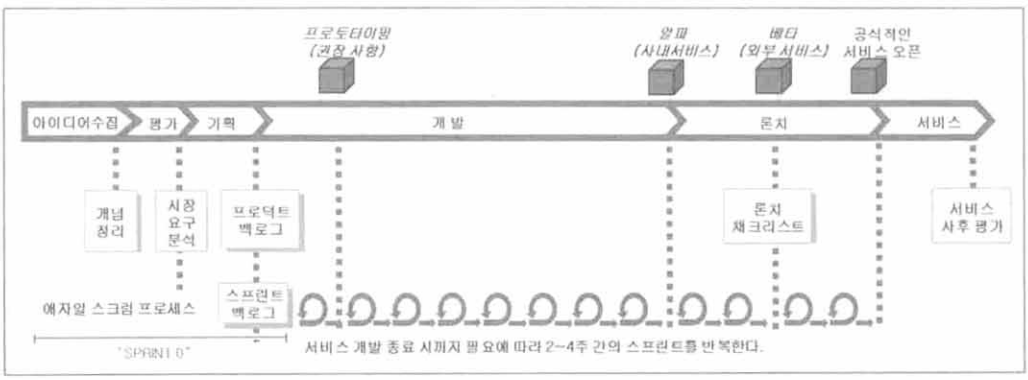
다. 반면, 경험이 풍부한 분산 애자일팀의 경우 적절한 완전 분산 스크럼 전략을 통해 일반 스크럼과 동일한 수준의 성능을 낼 수 있다[12, 13].

- 고립된 스크럼(Isolated Scrums): 여러 지역에 분산되어 있는 팀이 기능적으로 연결되어 있지 않거나 스크럼 프로세스를 사용하지 않는 경우.
- 분산 스크럼의 스크럼(Distributed Scrum of Scrums): 여러 지역에 분산되어 있는 팀이 스크럼의 스크럼[11]으로 통합되어 주기적으로 협업하는 경우.
- 완전 분산 스크럼(Fully distributed Scrums): 여러 지역에 분산되어 있는 팀이 기능적으로 밀접하게 연결되어 협업하는 경우.

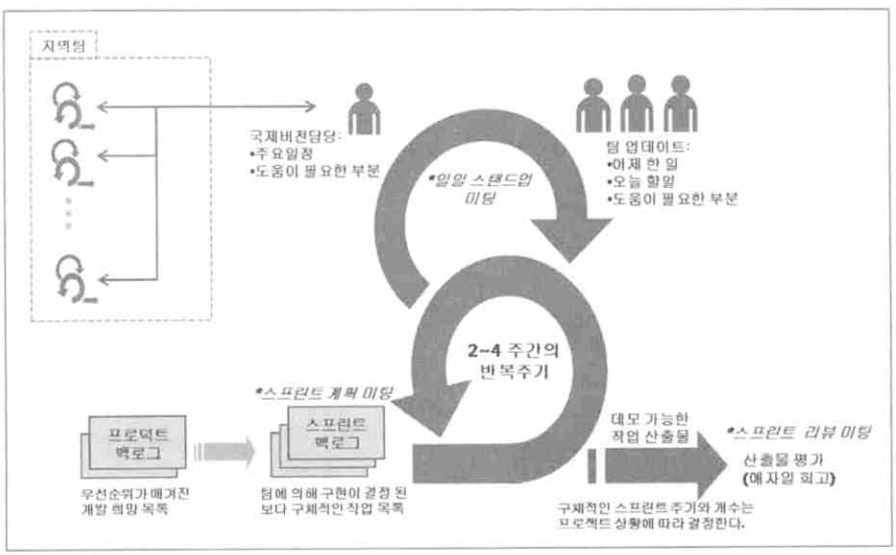
3.2 AFDSD 프로세스

본 논문이 제안하는 스크럼 프로세스는 2-4주 마다 반복되는 일련의 "스프린트(sprint)" 단위로 그 작업 결과가 점증적으로 출시된다(그림 2). 개발 단계 이전에는 프로젝트 전반에 걸친 아이디어 수집 및 개념 정리, 이를 바탕으로 한 시장조사 등을 통해 비전, 전략, 개발범위 등을 설정한다. "스프린트 0"으로 불리는 이러한 기획 및 준비 단계동안 고객의 요구사항을 프로덕트 백로그에 정리하고 본격적인 스프린트를 준비한다. 담당 프로덕트 매니저가 주관하는 프로덕트 백로그는 프로젝트가 종료될 때까지 지속적으로 새로운 요구사항이 추가되거나 기존 요구사항의 내용이 변경될 수 있는 가변적인 전체 태스크(task) 목록이다. 여기에 각 태스크 별 고유번호, 제목, 요약, 우선순위, 담당자, 분류, 초기 예상 작업 시간, 남아있는 작업 예상 시간, 가능한 수준의 명세, 관련 버그 정보 등이 기록 관리된다.

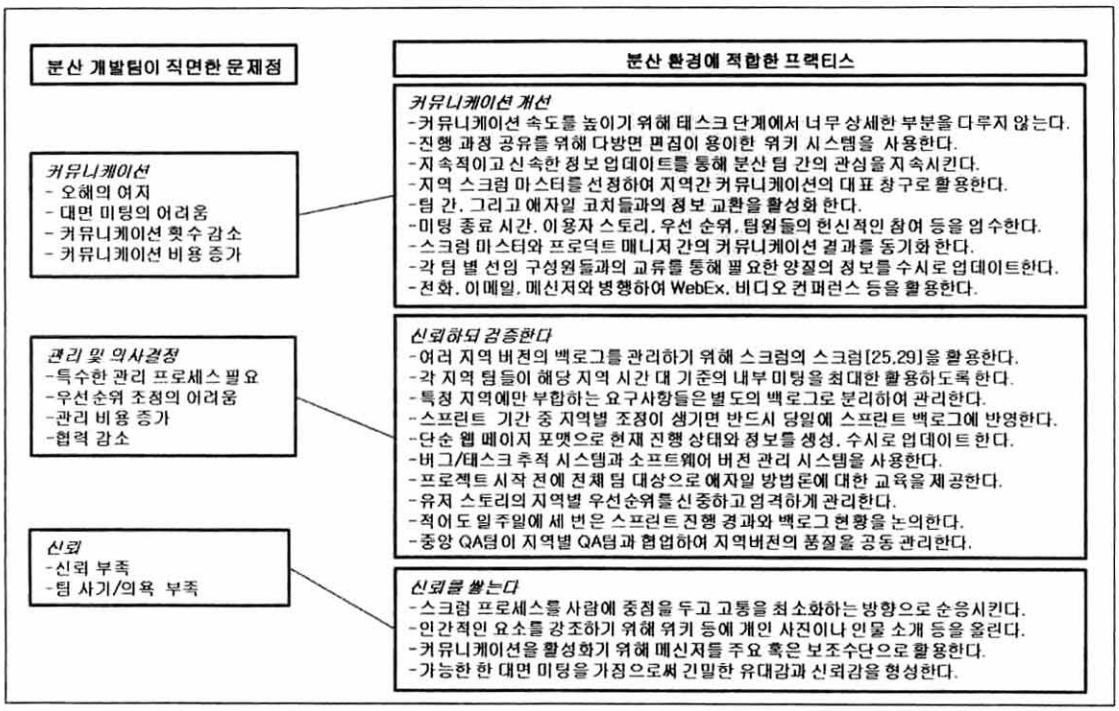
개발 단계에 들어가 스프린트가 시작되면 이전 스프린트 단계에서 준비된 실제 개발 태스크 목록인 스프린트 백로그를 기준으로 개발 및 테스트, 버그 수정 등이 이루어진다(그림 3). 스프린트 백로그는 프로덕트 백로그의 부분 집합이 구체화된 것이다. 매 스프린트 시작 전에 전체 개발팀이 참석하는 장시간의 스프린트 미팅을 통해 해당 스프린트에 대한 구체적인 목표와 범위, 개발 담당자, 가능한 작업 예상 시간 등이 결정된다. 일단 스프린트 백로그가 완성되고 나면 해당 스프린트가 종료될 때까지 원칙적으로 그 내용이



(그림 2) 스크럼에 기반한 AFDSD 프로세스



(그림 3) AFDSD 프로세스의 스프린트 반복 주기



(그림 4) 분산 개발팀이 직면한 문제점 vs. 프랙티스 매핑

변경될 수 없다.

3.3 AFDSD 프랙티스

AFDSD 프랙티스는 스크럼[25]과 XP 엔지니어링 프랙티스[26]의 결합을 기본으로 하여 그 민첩도를 극대화하였다 <표 1>. 그 밖에도 분산 개발 환경으로 기인한 어려움을 극복하기 위해 기존 사례 연구[6-9, 12, 13]를 바탕으로 (그림 4)와 같은 여러 프랙티스를 적용하였다.

4. 애자일 소프트웨어 분산 개발 프레임워크 (AFDSD) 적용 사례

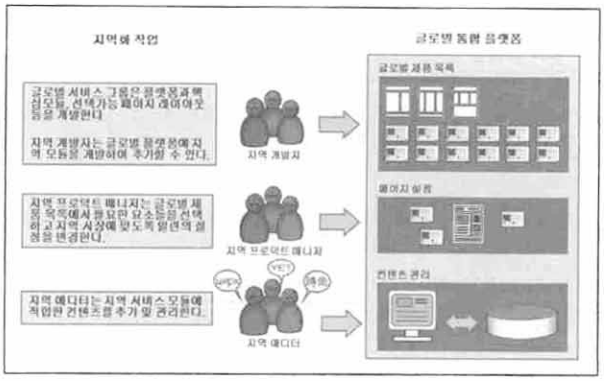
마이 야후는 미국을 중심으로 전 세계 약 20개 지역 버전이 제공되고 있는 개인화 홈페이지 서비스이다. 1996년 서비스 개시 이후 2008년까지 총 3개 버전의 플랫폼-클래식, 조로, 카멜레온-이 개발 및 세계화되었다. 본 연구에서는 이 가운데 최신 버전인 카멜레온 플랫폼의 개발 및 세계화 프로젝트에 AFDSD를 개발 초기부터 적용하여 성공적으로 구현하였다.

4.1 프로젝트 소개

소프트웨어의 세계화(globalization)란 국제화(internationalization)와 지역화(localization)를 포함하는 개념이다. 즉, 해당 제품의 플랫폼이 기술적으로 지역화가 가능하도록 코딩되어 있어야 하고(국제화), 이러한 플랫폼 위에 언어를 번역하고 해당 지역에 적합한 사회적, 문화적, 기능적 요소를 반영하여 (지역화), 세계의 어떤 이용자도 해당 서비스를 자신이 속한 지역사회의 서비스처럼 온전하게 이용할 수 있도록하는 일련의 작업을 의미한다[24]. 카멜레온은 이러한 기능을 제공하는 글로벌 통합 플랫폼이다. 그 개발 및 국제화 작업은 본사가 위치한 캘리포니아의 서니베일과 인도의 뱅갈로어에 분산된 본사 서비스 그룹이 주관하였다. 또한, 이 플랫폼을 이용한 일련의 지역화 작업은 한국, 대만, 홍콩, 싱가포르, 호주, 인도, 일본, 스페인, 영국, 프랑스, 이탈리아, 독일, 캐나다, 브라질, 멕시코, 아르헨티나 등에 위치한 각 지역팀이 주관하였다.

4.2 시스템 및 도구

카멜레온 플랫폼은 FreeBSD 6.x 위에 PHP(Personal Hypertext Preprocessor), C++, JS(JavaScript) 등을 사용하여 개발되었다. 카멜레온에서 제공되는 모듈의 상당 부분을 차지하는 RSS(Really Simple Syndication) 피드는 XSL(eXtensible Stylesheet Language), HTML(Hyper Text Markup Language), CSS(Cascading Style Sheets) 등을 이용하여 자체 개발된 피드 템플릿(feed template)에 의해 처리된다. 이보다 고급 기능의 모듈은 오픈 모듈(open module)이라고 하는 역시 자체 개발된 솔루션에 의해 구현되며 지역 개발자들에게는 JS 모듈 API(Application Programming Interface)가 제공되었다. 이 두 가지 방법으로 처리할 수 없는 복합적인 모듈은 포털 컴포넌트(portal component)를 이용하여 개발할 수 있다. 이 부분은 플랫폼에 직접 복잡한 PHP 코드를 추가해야 하므로 주로 본사 서비스 그룹이 작업을 주관하였다. 대부분의 경우, 지역 개발자는 지정된 스펙에 맞추어 피드 템플릿을 변경하거나 플랫폼이 제공하지 않는 새로운 모듈이 필



(그림 5) 강력한 지역화 기능을 제공하는 카멜레온 플랫폼

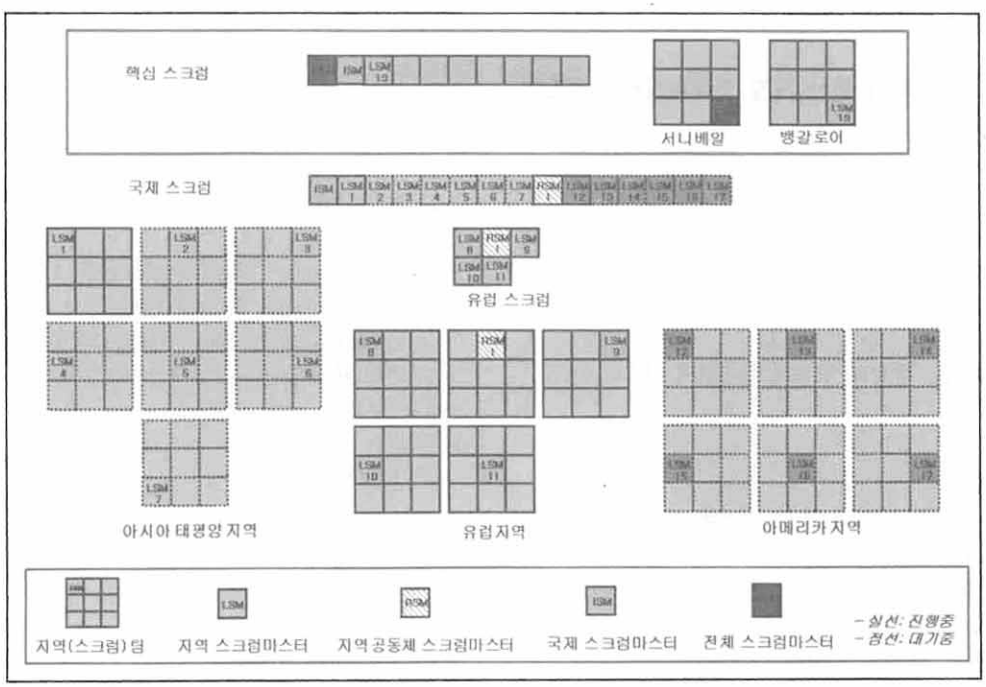
요할 경우 오픈모듈 API를 이용하여 지역 모듈을 개발 및 추가하였다. 한편 지역화 프로젝트의 PM인 각 지역 프로덕트 매니저는 플랫폼 지역화 도구를 이용하여 번역을 포함, 해당 지역 시장에 필요한 일련의 설정을 그에 적합하도록 세팅하였다. 이렇게 지역 버전의 서비스 틀이 완성되고 나면 지역 에디터 혹은 프로덕트 매니저가 해당 지역 컨텐츠를 지속적으로 추가 및 관리한다(그림 5).

4.3 소프트웨어 분산 개발 조직

카멜레온 프로젝트는 분산 스크럼 기반의 AFDSD 개발 조직모델(3.1)을 사용하였다. 서니베일과 뱅갈로어에 분산되어 있는 본사의 제품 개발 그룹과 영국, 프랑스, 스페인, 이탈리아, 독일 등에 분산되어 있는 유럽지역 그룹은 완전 분산 스크럼 전략을 사용하였다. 국제 지역 담당자의 주관 아래 여러지역팀과 본사 서비스 그룹과의 협업으로 이루어지는 지역화 작업은 분산스크럼의 스크럼 전략에 의해 구현되었다. 예외적으로 본사의 소스 코드를 가져다가 자체적인 단일 플랫폼을 구축하여 독자적으로 서비스를 운영하는 일본팀의 경우가 고립된 스크럼의 예이다. 카멜레온에서 사용한 복합 분산 스크럼은 (그림 6)과 같이 4 종류의 스크럼으로 구성되었다.- 1) 핵심팀과 본사 서비스 그룹이 협업하는 핵심 스크럼, 2) 국제 버전 담당자, 각 지역팀 그리고 유럽 연합 그룹이 협업하는 국제 스크럼, 3) 유럽 연합 팀 간에 협업하는 유럽 스크럼, 4) 아시아 태평양과 아메리카 대륙의 각 지역이 독자적으로 구성한 지역 스크럼. 이러한 구조는 국제 스크럼을 중심으로 지역 서비스의 체계적인 중앙 관리가 가능한 장점이 있다. 반면, 한번에 많은 수의 지역화 작업을 적극적으로 지원하기에는 본사 서비스 그룹 리소스의 한계가 있었다. (그림 6)에서 점선의 스크럼은 본사로부터 지역화 작업 지원을 받기 위해 대기 중으로, 카멜레온의 지역화 작업이 한번에 3-7개 정도만 동시에 이루어질 수 있었던 한계점을 반영한다.

4.4 소프트웨어 분산 개발 프로세스 및 프랙티스

카멜레온 프로젝트는 AFDSD의 프로세스(3.2)와 프랙티스(3.3)를 기반으로 진행 및 완료되었다. 개발자는 각자의 개발서버를 구축하여 할당 된 태스크를 개발하였다. 해당 태스크가 완성되면 전 세계 개발자가 공동으로 사용하는 알파 서버에서 정상적인 동작 여부를 통합 테스트했다. 이렇게 개발자를 위한 알파 릴리즈는 하루에 한번, 실제 사내 통합 테스트의 대상이 되는 베타릴리즈는 일주일에 두 번 이루어졌다. 해당 스프린트 기간 동안의 작업 내용 전체가 베타 서버에서 무리없이 작동하면 2주에 한번씩 스테이징



(그림 6) 카멜레온 프로젝트에서 사용된 복합 분산 스크럼 전략

서버로 릴리즈되어 QA팀의 최종 단계 평가대상이 된다. 여기서 통과되면 실제 프로덕션 서버로 릴리즈되어 전 세계 이용자들에게 서비스 되었다.

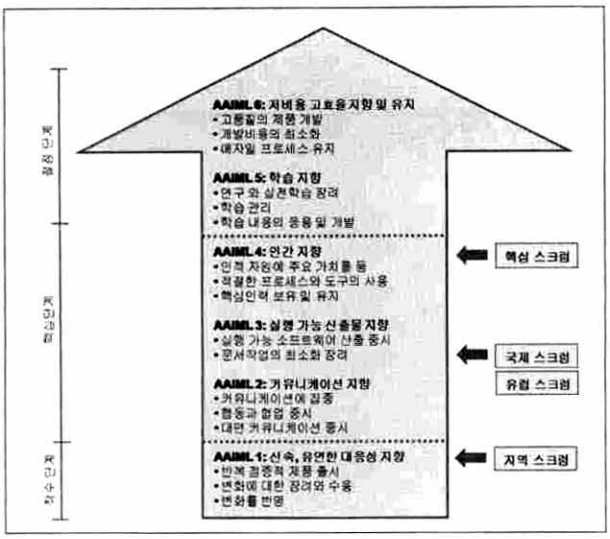
한편, 전체 스크럼 마스터가 주관하는 매일 아침 15분 간의 스탠드업 미팅을 통해 개발자 전원이 어제의 작업 내용, 오늘의 작업 계획, 작업 중에 도움이 필요한 부분 등을 짤막하게 공유하여 개개의 업무 처리 및 상호 협력 체계를 효율화하였다. 이후 매일 관련이 있는 핵심팀 혹은 개발팀의 담당자를 대상으로 30분 간의 일일버그 처리 미팅이 이어졌다. 여기서 사내 버그/태스크 관리 시스템에 올라온 당일의 버그 및 태스크에 처리 우선 순위, 담당자, 처리 완료 목표 시점 등이 도움이 될만한 간략한 정보와 함께 할당되었다. 이 두 가지 일일 미팅은 특히 국제 버전 담당자가 팀 전원을 매일 한자리에 만나 각 지역 버전 서비스가 직면하고 있는 긴급한 문제에 대한 본사 팀의 협조를 신속하게 얻어 내는데 중요한 역할을 했다.

5. 애자일 소프트웨어 분산 개발 프레임워크 (AFDSD) 평가

AFDSD는 개발 조직 모델, 프로세스 및 프랙티스로 구성된다. 본 논문에서는 AFDSD의 효율성을 입증하기 위해 다음과 같은 4가지 척도를 이용하여 그 구성요소 및 적용 결과를 평가하였다. 첫째, 적용사례에서 사용된 개발 조직의 애자일 도입 수준을 평가하여 AFDSD의 개발 조직 모델이 성공적으로 적용됨을 보였다. 둘째, 적용사례에서 사용된 AFDSD 프로세스와 프랙티스에 대한 민첩도를 산출하고 이를 기존 애자일 방법론의 경우와 비교함으로써 AFDSD가 민첩성 측면에서 기존의 애자일 방법론보다 우수함을 보였다. 셋째, 애자일 도입 수준 별로 적용 사례를 세분하여 각 애자일 프로젝트 성공도를 산출 및 비교함으로써 애자일 도입 정도에 따라 프로젝트 성공 가능성에 차이가 있음을 보였다. 넷째, AFDSD 하에서 개발된 함수 개수, 주요 버그 수 등을 분산 폭포수 프로세스 하에서 개발된 이전 버전과 비교하여 그 성능과 품질의 개선 정도를 보였다.

5.1 개발 조직의 애자일 도입수준

본 절에서는 카멜레온 프로젝트의 개발 주체인 4 종류의 스크럼(4.3)에 대해 애자일 도입 및 개선 단계[22]를 평가하였다(그림 7). 카멜레온 플랫폼의 개발 및 국제화 작업을 주도한 핵심 스크럼은 전문 핵심 인력 중심의 의사 결정 체계, 프로세스, 도구 등이 정착되었으므로 AAIML 4에 포함되었다. 반면, 서비스의 지역화 작업을 주도한 대다수의 지역 스크럼의 경우 핵심 스크럼의 영향 아래 신속, 유연, 변화의 적극 수용이라는 애자일의 기본 원칙에 충실하는 단계인 AAIML 1에 머물러 있다. 그 중간에 위치하는 국제 스크럼과 유럽 스크럼은 적극적인 상호 협력과 커뮤니케이션을 추구하며 AAIML 2에서 AAIML 3으로 개선하고 있다. 본 평



(그림 7) 카멜레온 프로젝트의 개발 주체 별 애자일 도입 수준

가를 통해 기존에 애자일의 도입이 활발하게 이루어지고 있던 북미와 유럽지역의 경우 상대적으로 AFDSD가 용이하게 적용됨을 알 수 있었다

5.2 프로세스와 프랙티스의 민첩도

본 절에서는 4-DAT[23]를 이용하여 카멜레온 프로젝트에 적용된 AFDSD 프로세스와 프랙티스의 민첩도를 평가하였다. 민첩도는 0부터 1 사이의 값으로 숫자가 클수록 민첩성의 정도가 높으며, 이를 판단하기 위한 threshold는 0.5에서 0.6 사이이다. 4-DAT를 이용하여 소프트웨어 개발 방법론에 대한 민첩도*(DA(Method))*를 평가하기 위해 해당 방법론에서 사용된 프로세스의 구성 단계 및 프랙티스의 민첩도, *DA(Method, Phases)*와 *DA(Method, Practices)*를 각각 산출한다. 즉, 각 프로세스 구성 단계 및 프랙티스의 유연성 (flexibility: FY), 신속성 (speed: SD), 경제성 (leanness: LS), 학습효과 (learning: LG), 대응성 (responsive: RS) 여부를 이진법으로 측정하여 모두 합산하고 전체 항목 수(m)로 나누어 구한 값을 다음과 같이 계산한다.

$$DA(Method) = \frac{1}{m}\sum mDA(Method, Phases \text{ or } Practices)$$

본 논문에서 *DA(AFDSD, Phases)*와 *DA(AFDSD, Practices)*를 산출한 결과, 각각 0.8과 0.83이 나왔다<표 1, 2>. 이를 기존의 애자일 방법론에 대한 평가 결과[23]와 비교하면, 프로세스의 경우 가장 민첩도가 높은 크리스탈 패밀리(0.8)와 동일하고, 프랙티스의 경우는 가장 민첩도가 높은 스크럼(0.8)보다 높은 수치이다. 여기서 산출된 각 프로세스와 프랙티스 값을 이용하여 *DA(AFDSD)*를 구하면 0.815로 기존의 애자일 방법론 가운데 민첩도가 높은 순서인 크리스탈 패밀리, XP, 스크럼에 비해 각각 6%, 12%, 14% 향상되었음을 알 수 있다.

〈표 1〉 카멜레온 프로젝트에 적용된 AFDSD 민첩도 평가

카멜레온	민첩도 속성					
	FY	SD	LS	LG	RS	총계
프로세스 구성 단계(Phases)						
스프린트 0	1	1	0	1	1	4
개발	1	1	0	1	1	4
론치	1	1	0	1	1	4
총계	3	3	0	3	3	12
DA(AFDSD, Phases)	3/3	3/3	0/3	3/3	3/3	12/(3*5)
프랙티스(Practices)						
프로덕트 백로그	1	1	0	1	1	4
스프린트 백로그	1	1	0	1	1	4
코딩 표준	1	1	1	1	1	5
정기적인 빌드	1	1	0	1	1	4
Configuration 관리	1	1	0	1	1	4
변경 내용 원상복귀 지원	1	1	0	1	1	4
지속적인 코드 통합	1	1	1	1	1	5
테스트 우선 프로그래밍	1	1	0	1	1	4
통합 테스트	1	1	0	1	1	4
스프린트 리뷰	1	1	0	1	1	4
스크럼 팀	1	1	0	1	1	4
스크럼 마스터	1	1	0	1	1	4
스프린트 계획 미팅	1	1	0	1	1	4
총계	13	13	2	13	13	54
DA(AFDSD, Practices)	13/13	18/20	2/13	13/13	13/13	54/(13*5)

〈표 2〉 AFDSD와 기존 애자일 방법론[23]과의 민첩도 비교

방법론	*DA(Phases)*	*DA(Practices)*	*DA(Method)*
XP	0.702	0.732	0.717
스크럼	0.603	0.800	0.702
FDD	0.485	0.703	0.594
DSDM	0.466	0.684	0.575
크리스탈	0.800	0.732	0.766
AFDSD	0.800	0.830	0.815

5.3 애자일 프로젝트 성공도

본 절에서는 애자일 프로젝트 성공 요소 및 그 속성<표 3>에 대한 연구[21]를 기반으로 애자일 프로젝트 성공도 *(DAPS(Project))*를 계산하기 위한 공식을 개발하고 최초로 그 값을 산출하였다. 기존 연구에서 제시한 주요 성공 요소(Critical Success Factors: CFS)와 각 성공 요소 별 속성 및 중요도(가중치)를 이용하여 특정 프로젝트에 대한 개별 성공요소의 성취도*(DAPS(Project, CSF))*를 계산하고, 이를 모두 합한 값이 해당 프로젝트의 성공도*(DAPS(Project))*가 된다. 애자일 프로젝트 성공도는 0부터 1 사이의 값으로 숫자가 클수록 프로젝트 성공 가능성이 높아지며, 이를 판단하기 위한 threshold는 0.7부터 0.8 사이가 적합하다. 본 논문에서 개발한 아래의 공식은 기존 연구는 물론, 주요 성공 요소 및 그 속성의 추후 확장 가능성을 수용할 수 있도록 유연하게 설계되었다.

〈표 3〉 애자일 프로젝트 성공 평가를 위한 6가지 주요 요소 및 그 속성[21]

순위	주요 성공 요소	속성
1	제품 출시 전략	DS1: 정기적인 소프트웨어 출시
		DS2: 가장 중요한 기능을 먼저 출시
2	애자일 소프트웨어 엔지니어링 기술	AT1: 잘 정의된 코딩 표준 적용
		AT2: 간결한 설계 추구
		AT3: 엄격한 현존 코드의 개선(refactoring)
		AT4: 적절한 정도와 분량의 문서화
		AT5: 올바른 통합 테스트
3	팀 역량	TC1: 높은 자신감과 전문지식을 가진 팀원들
		TC2: 높은 성취 동기를 가진 팀원들
		TC3: 애자일 기술에 대한 지식과 경험이 풍부한 관리자
		TC4: 현장 적응력이 높은 관리자
		TC5: 팀에 대한 적합한 기술 교육 제공
4	프로젝트 관리 프로세스	PM1: 애자일 지향적인 요구사항 관리 프로세스 준수
		PM2: 애자일 지향적인 프로젝트 관리 프로세스 준수
		PM3: 애자일 지향적인 configuration 관리 프로세스 준수
		PM4: 적절한 프로젝트 진행사항 추적 기법
		PM5: 일일 대면 미팅을 통한 강력한 커뮤니케이션
		PM6: 정기적인 작업 스케줄 준수
5	팀 환경	TE1: 전체 팀이 같은 작업 공간에 위치
		TE2: 밀접하고 자발적인 팀
		TE3: 소규모 팀으로 구성된 프로젝트
		TE4: 독립적인 여러 팀과 공동 수행하지 않는 프로젝트
6	고객 참여	CI1: 고객과 우호적인 관계
		CI2: 강력한 고객의 지원 및 직접적인 참여
		CI3: 문제 해결에 대한 권위를 가진 고객

$$DAPS(Project) = \sum DAPS(Project, CFS)$$
$$CFS = DS, AT, TC, PM, TE, CI$$

$$DAPS(Project, CFS) = W \times (T/N)$$
(W: 가중치, N: 속성의 개수, T: 속성값의 총합)

효과적인 애자일 프로젝트의 성공도 평가를 위하여 본 절에서는 카멜레온 프로젝트를 각 지역버전의 지역화 프로젝트로 세분하였다. 즉, 그 개발 주체의 애자일 도입 수준에 따라 3 그룹-1) AAIML 4의 미국 버전, 2) AAIML 1이상의 애자일 지역 버전, 3) 지역 여건 상 애자일의 도입이 제대로 이루어지기 어려운 비애자일 지역 버전-으로 분류하였다. 이와 같은 조건 하에 *DAPS(Chameleon)*을 산출한 결과, 애자일 도입 수준이 가장 높은 미국 버전의 경우 0.95, 애자일 기법을 적극 도입하여 지역화 작업을 수행한 애자일 지역

〈표 4〉 카멜레온 프로젝트 성공도 - 미국, 애자일 지역 및 일반 지역 버전 비교

가중치(W)	주요 성공 요소 (CSF)	속성	카멜레온 프로젝트		
			미국	애자일 지역	비애자일 지역
0.3	제품 출시 전략 (DS)	DS1	1	1	1
		DS2	1	1	1
		총계	2	2	2
0.2	애자일 소프트웨어 공학 기술 (AT)	AT1	1	1	1
		AT2	1	1	1
		AT3	1	0	0
		AT4	1	1	0
		AT5	1	1	1
		총계	5	4	3
0.2	팀 역량 (TC)	TC1	1	1	0
		TC2	1	1	0
		TC3	1	1	0
		TC4	1	1	1
		TC5	1	1	0
		총계	5	5	1
0.1	프로젝트 관리 프로세스 (PM)	PM1	1	1	1
		PM2	1	1	1
		PM3	1	1	1
		PM4	1	1	0
		PM5	1	0	0
		PM6	1	1	1
		총계	6	5	4
0.1	팀 환경 (TE)	TE1	0	0	0
		TE2	1	1	0
		TE3	1	1	1
		TE4	0	0	0
		총계	2	2	1
0.1	고객 참여 (CI)	CI1	1	1	1
		CI2	1	1	1
		CI3	1	1	1
		총계	3	3	3
DAPS(Chameleon)			0.95	0.89	0.65

버전의 경우 0.89로 역시 높은 수치가 나왔다. 반면, 동일한 플랫폼과 도구, AFDSD 기반의 본사와의 협력 체계 하에서도 비애자일 지역 버전 출시에 대한 프로젝트 성공도는 평균적으로 0.65가 나왔다. 이는 적극적으로 애자일 기법을 도입한 지역 버전보다 27% 낮은 수치이다<표 4>. 카멜레온 서비스의 지역화 작업을 효과적으로 수행하기 위해 이와 같은 일부 비애자일 지역의 경우 지역화 작업의 주도권을 해당 지역팀이 아닌 본사 서비스 그룹의 직접 관리 아래 아웃소싱 인력 등을 활용하여 수행하였다. 이처럼 애자일 프로젝트 성공도는 프로젝트 수행 전 성공 가능성을 예측하는 도구로 사용될 수 있다.

5.4 생산성과 품질

2007년부터 2008년에 걸쳐 출시된 카멜레온 지역 버전의 경우, 평균적으로 지역팀과 본사 서비스 팀 3명이 약 5개월간의 작업 기간을 필요로 했다. 이는 약 2년 전 분산 폭포수 방식 하에서 같은 인원이 약 7개월을 필요로 했던 이전 버전(조로)에 비해 괄목할만한 개선 결과이다. 개발자가 한 달 동안 완성한 함수의 평균 개수와 지역화 작업 기간 동안 보고된 주요 버그의 개수를 각각 플랫폼의 생산성과 품질의 척도라고 보자. 이러한 가정 하에, 카멜레온 버전의 경우 조로 버전과 비교하여 그 생산성과 품질이 각각 40%, 36% 향상되었다<표 5>. 각기 지역화 업무 범위가 다른 두 개의 플랫폼을 평면적으로 비교하는 것은 무리가 있으나, 그 밖에도 2008년 초에 실시했던 비공식적인 사내 설문조사에 의하면 이전 버전 대비 새로운 플랫폼과 그 개발 및 지역화 프로세스에 대해 참여자의 대부분이 평균 30% 이상의 개선효과를 보았다고 응답했다.

〈표 5〉 마이 야후 지역화 작업 결과에 대한 평균 생산성 및 품질 비교

버전	프로세스	출시 기간	man/month	함수 개수 (man/month)	주요 버그 개수
조로	분산 폭포수	2005-2006	21	12.5	108
카멜레온	AFDSD	2007-2008	15	17.5	39

5.5 평가에 대한 논의

애자일을 도입한 업체는 대부분 개발팀과 관리자의 개별 피드백 등을 통해 단기간에 주목할만한 생산성의 향상을 발견한다. 하지만 이를 객관적이고 과학적인 방법으로 측정하기란 쉽지 않다[2, 11, 18, 19]. 기존 연구를 살펴보면 애자일 도입 전과 후에 산출 된 총 프로그램 코드 라인 수, 혹은 함수의 총 개수 등을 비교하여 그 성능 개선의 정도를 평가[12, 13, 20]하는 방법을 주로 사용하고 있지만 이는 개발 속도만을 평면적으로 비교하는 것이므로 한계가 따른다. 많은 애자일 기업이 설문조사[1-3, 10, 11]에 의존하여 애자일 도입효과를 분석하는 것도 같은 맥락이다.

본 논문에서는 이렇듯 평가 측면에서 보여지는 애자일 프로젝트의 약점을 보완하기 위해 다양한 관점에서 평가를 수행하였다. 애자일 도입 수준 평가를 통해 AFDSD 개발 조직 모델의 적응성을 보였으며, 민첩도 산출을 통해 AFDSD의 프로세스와 프랙티스의 민첩성이 우수함을 보였다. 또한, 애자일 프로젝트 성공도 산출을 통해 AFDSD 적용 프로젝트의 성공 가능성을 예측할 수 있음을 보였으며, 개발 완료 후 실제 데이터를 이용한 생산성 및 품질의 개선 정도를 보였다.

분산 스크럼의 성능평가를 수행한 2건의 기존 연구에서 자바 개발자가 한달 동안 개발한 함수의 개수는 SirsiDynix사[12]의 경우 15.3, Xebia사[13]의 경우 15.1이며, 이는 일반 스크럼[20, 12]의 경우인 17.8과 유사하다. 개발 속도만 평면적으로 놓고 보면 평균 17.5개의 함수를 개발한 카멜레온의 경우 그 성능이 우수하다고 평가할 수있다<표 5>. 그러나 실제 개발 환경 및 작업 난이도의 차이 등을 반영하여 비교하기에는 한계가 따른다. 또한, 관련 최신 연구를 바탕으로 앞서 평가한 애자일 도입 수준, 민첩도, 애자일 프로젝트 성공도 등의 산출 기준 역시 추후 확장 및 보완의 여지가 있다는 점을 감안해야 한다.

AFDSD의 성공 요인은 애자일 소프트웨어 분산 개발의 최적화를 위해 민첩도가 높은 프로세스와 프랙티스를 선별하여 적용한 데에 있다고 판단된다. 민첩도가 높은 소프트웨어 개발 방법론은 인간 중심적, 커뮤니케이션 지향적이며, 시장의 변화에 유연하게 대응한다. 또한, 짧은 주기의 반복 점증적인 릴리즈를 통해 저비용 고효율과 지속적인 개선을 추구함으로써 비즈니스의 성공을 지향한다. 더불어, 글로벌 분산 환경의 약점을 최소화하기 위해 소프트웨어 분산 개발을 위한 프랙티스(그림 4)를 적극 활용하였다.

6. 결 론

애자일 방법론은 대면 커뮤니케이션, 자발적인 소규모의 팀, 협동, 변화하는 요구사항, 짧은 출시 주기, 우수한 기술력, 애자일 회고에 의한 지속적인 개선 등을 강조하며 오늘날 직면한 소위 소프트웨어 위기를 극복하고 있다[1-3]. 이렇듯 직접적인 커뮤니케이션을 통한 의사소통을 핵심으로 하는 애자일은 소프트웨어 분산 개발 분야에서도 최근 그 진가를 발휘하고 있다[6-9]. 격주로 온라인 협의를 거쳐 합리적인 양의 요구 사항을 공유하고, 매일 일정 시간씩 진행 상황을 간략히 상호 점검하는 것이 당장 가시적인 성과로 연결되기 때문이다. 커뮤니케이션의 장벽이 높을수록 업무에 대한 피드백 주기를 짧게 하고 그 결과를 더욱 자주 평가한다는 상식은 이미 전 세계 애자일 글로벌 기업에 의해 입증되고 있다[1-3, 10-13]. 더불어 담당 상위 관리 그룹의 적극적인 지원, 프로덕트 기획팀과 개발팀의 헌신적인 참여와 협동은 성공적인 소프트웨어 개발 및 세계화를 위한 필수 항목이다[24]. 애자일은 프로젝트 가시성이 높아 현업에서 부담으로 작용할 수도 있다. 하지만 이를 사용해본 개발팀의 과반수 이상이 지속적인 사용을 원하는 이유는, 사람이 기본적인 개발 주체인 소프트웨어 프로젝트에 엄격하고 한정된 프로세스를 적용하기보다는 변화의 가능성을 열어두고 환영하는 애자일 철학이 비즈니스 성공과 더불어 개인 능력의 향상에도 큰 도움을 주기 때문이다[10, 11].

본 논문에서는 애자일 소프트웨어 분산 개발에 관한 최신 연구를 바탕으로 분산 스크럼 전략과 최적화된 애자일 기법을 사용하여 AFDSD를 설계하였으며, 이를 미국 야후사의 카멜레온 프로젝트에 실제 적용하여 성공적으로 구현하였다. 또한, 개발 조직의 애자일 도입 수준, 프로세스와 프랙티스의 민첩도, 애자일 프로젝트 성공도 및 생산성과 품질 개선 측면에서 본 적용사례를 종합적으로 평가함으로써 AFDSD의 효율성을 입증하였다. 특히, 애자일 프로젝트 성공도는 관련 최신 연구를 바탕으로 본 논문에서 그 공식을 최초로 개발 및 적용하였다. 본 연구의 목적은 국제화 시대에 글로벌 소프트웨어 분산 개발팀이 직면한 도전을 고찰하고 애자일을 그에 대한 현실적인 대안으로 제안하는 것이다. 본 논문에서 제시한 개발 조직 모델, 분산 스크럼 전략, 애자일 스크럼 프로세스, 애자일 소프트웨어 분산 개발 프랙티스 및 평가 모델 등은 실제 산업 현장에서 곧바로 활용될 수 있다고 판단된다.

7. 감사의 글

지난 4년간 마이 야후 서비스의 개발 및 세계화를 위해 함께 애써주신 서니베일과 뱅갈로어에 위치한 글로벌 서비스 그룹 이하 전 세계 3개 대륙 여러 지역팀(한국, 대만, 홍콩, 싱가포르, 인도, 호주, 일본, 중국, 캐나다, 스페인, 영국, 프랑스, 이탈리아, 독일, 브라질, 멕시코, 아르헨티나, 마이애미) 여러분께 진심으로 감사의 마음을 전합니다.

참 고 문 헌

[1] O. Salo, P. Abrahamsson, "Agile Methodsin European Embedded Development Organizations: a survey study of Extreme Programming and Scrum," IET Software, Vol.2, pp.58-64, 2008.

[2] T. Dyba˚, T. Dingsøyr, "Empirical studies of agile software development: A systematic review," Information and

Software Technology, Vol.50, No.9-10, pp.833-859, 2008.
[3] C. Schwaber, "Enterprise Agile Adoption In 2007," Forrester Research, Feb., 2008.
[4] 강규영, "개발 환경에서 본 애자일 :오픈마루 개발자의 도입 사례 소개", 마이크로소프트웨어, 통권281호, pp.178-183, 2007.
[5] 김창준, "오늘의 모습으로 내다본 애자일의 미래", 마이크로소프트웨어, 통권281호, pp.184-189, 2007.
[6] P. Ågerfalk, B. Fitzgerald, "Flexible and distributed software processes: old petunias in new bowls," Communications of the ACM, Vol.49, No.10, pp.27-34, 2006.
[7] G. Lee, W. DeLone, J. A. Espinosa, "Ambidextrous coping strategies in globally distributed software development projects," Communications of the ACM, Vol.49, No.10, pp.35-40, 2006.
[8] B. Ramesh, L. Cao, K. Mohan, P. Xu, "Can distributed software development be agile?," Communications of the ACM, Vol.49, No.10, pp.41-46, 2006.
[9] N. Jain, J. Eckstein, "Distributed agile stage," In: Agile 2008, Available online at http://www.agile2008.org/stage-distributed.html, 2008.
[10] A. Begel, N. Nagappan, "Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study," First International Symposium on Empirical Software Engineering and Metrics, 2007.
[11] J. Sutherland, K. Schwaber, 'The scrum papers: nuts, bolts, and origins of an agile method,' Scrum Inc., 2007.
[12] J. Sutherland, A. Viktorov, J. Blount, N. Puntikov, "Distributed Scrum: agile project management with outsourced development teams," In:Hawaii International Conference on Software Systems (HICSS'40), Big Island, Hawaii, 2007.
[13] J. Sutherland, G. Schoonheim, E. Rustenburg, M. Rijk, "Fully distributed scrum: the secret sauce for hyperproductive offshored development teams". In:Agile Conference 2008, pp.339-344, 2008.
[14] J. Highsmith, 'Agile software development ecosystems,' Addison-Wesley, 2002.
[15] K. Beck, M. Beedle, A. Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas (2001), 'Manifesto for agile software development', Available at: http://agilemanifesto.org/.
[16] R. Hira R, "Testimony to the US-China economic security review commission on offshoring of software & high technology jobs," IEEE-USA, Available online at www.ieeeusa.org/policy/POLICY/2005/021305.pdf, 2005.
[17] B. Fitzgerald, G. Hartnett, K. Conboy, "Customizing agile methods to software practices at intel Shannon," European Journal of Information Systems, Vol.15, No.2, pp.200-213, 2006.
[18] P. Abrahamsson, O. Salo, J. Ronkainen,J. Warsta, 'Agile software development methods: review and analysis,' VTT Technical report, 2002.
[19] P. Mcbreen, 'Questioning extreme programming,' Pearson Education, ISBN 0-201-84457-5, 2003.
[20] M. Cohn, 'User stories applied: for agile software development,' Addison-Wesley, 2004.
[21] T. Chow, D. Cao, "A survey study of critical success factors in agile software projects," The Journal of Systems and Software, Vol.81, pp.961-971, 2008.
[22] A. Qumer, B. Henderson-Sellers, "A framework to support the evaluation, adoption and improvement of agile methods in practice," The Journal of Systems and Software, Vol.81, 1899-1919, 2008.
[23] A. Qumer, B. Henderson-Sellers, "An evaluation of the degree of agility in six agile methods and its applicability for method engineering," Information and Software Technology, Vol.50, pp.280-295, 2008.
[24] E. Flarup, "Best practices in software localization," In: Internationalization & Unicode Conference (IUC32), San Jose, California, 2007.
[25] K. Schwaber, M. Beedle, 'Agile Software Development with Scrum,' Prentice Hall, 2002.
[26] K. Beck, 'Extreme programming explained: embrace change,' Addison-Wesley, 2000.

이 세 영

e-mail : sarahlee230@yahoo.com
1995년 동덕여자대학교 전자계산학과(학사)
1999년 이화여자대학교 컴퓨터정보학과 (공학석사)
2002년~현 재 이화여자대학교 컴퓨터공학과 박사과정
1995년~2000년 ㈜나우콤 서비스 개발 연구소 전임연구원
2000년~2001년 ㈜데이콤 기술연구소 선임연구원
2001년~2005년 야후코리아 기술본부 기술매니저
2005년~2008년 Yahoo! Inc. Front Door Group, Sr. Program Manager
관심분야 : 애자일 방법론, 소프트웨어 개발 방법론, 소프트웨어 개발 관리, 소프트웨어 국제화/지역화

용 환 승

e-mail : hsyong@ewha.ac.kr

1983년 서울대학교 컴퓨터공학과(학사)

1985년 서울대학교 컴퓨터공학과(공학석사)

1985년~1989년 한국전자통신연구소 연구원

1994년 서울대학교 컴퓨터공학과(공학박사)

1994년 서울대학교 컴퓨터신기술공동연구소 특별연구원

1995년~현 재 이화여자대학교 컴퓨터공학과 교수

관심분야 : 데이터 마이닝, 멀티미디어 데이타베이스, 온톨로지 정보 검색